### **Job Scheduling Documentation for the OpenWeather ETL Pipeline**

The job scheduling process for the OpenWeather ETL pipeline was implemented using **Apache Airflow** to automate the extraction, transformation, and loading (ETL) of real-time weather data into an AWS S3 bucket. Below is an overview of how the scheduling was configured:

1. **DAG Creation and Scheduling**:
   * A Directed Acyclic Graph (DAG) was created to orchestrate the ETL tasks. The DAG was defined to run at specific intervals, making use of the Airflow schedule\_interval parameter.
   * In this project, the pipeline was scheduled to run **hourly**. The cron expression "0 \* \* \* \*" was used, meaning the pipeline would trigger at the beginning of every hour.
   * The DAG was designed to include tasks for extracting weather data from the OpenWeather API, transforming it, and loading the final output into the S3 bucket.
2. **Task Definition and Dependencies**:
   * Multiple tasks were defined within the DAG, such as:
     + Extracting data from the OpenWeather API using Python scripts.
     + Cleaning the data by handling missing values and duplicates.
     + Uploading the cleaned data to an S3 bucket.
   * Dependencies between the tasks were set to ensure that the pipeline follows the correct sequence, ensuring data extraction occurs before transformation and loading.
3. **Heartbeat and Monitoring**:
   * The Airflow scheduler was set up to monitor the DAG for any changes or errors. It was configured to check the DAG directory at regular intervals, triggering the pipeline according to the defined schedule.
   * The scheduler's heartbeat was monitored to ensure it was continuously running, and any interruption was logged and addressed by checking the Airflow logs using the ps aux | grep airflow command.
4. **Airflow Sensors**:
   * In addition to the regular scheduling, Airflow sensors were used to ensure the availability of the required API data before initiating the ETL process. This sensor checked the API's response and only triggered the pipeline when the data was available.
5. **Error Handling**:
   * Airflow’s retry mechanism was enabled for tasks in case of temporary failures such as network issues or API downtime. This ensured robustness and minimized the risk of data loss during the ETL process.
6. **Updating the Schedule**:
   * The schedule can be easily modified by adjusting the schedule\_interval parameter within the DAG definition. For example, changing the interval to "@daily" would trigger the pipeline once per day at midnight.
   * Any updates to the schedule required restarting the Airflow scheduler to ensure that the new intervals were applied.
7. **Final Output**:
   * Once the pipeline ran successfully, the transformed weather data was loaded into the designated S3 bucket, which could then be accessed for further analysis or reporting.